(72) Inventors:
• **Furuya, Soichi**
Cambridge CB1 3LR (GB)
• **Roe, Michael**
Cambridge CB1 3LR (GB)

(74) Representative: **Read, Matthew Charles et al**
Venner Shipley & Co.
20 Little Britain
London EC1A 7DH (GB)

(54) **Cryptographic apparatus and method**

(57) An encryption system comprises a pseudo-random number generator (KS) for generating a long pseudo-random sequence (S) from a shorter encryption key (K) and, if necessary, a nonce value (N), and a mixing function (MX) for combining the sequence with a plaintext message (P) on a block-by-block basis, where successive blocks (S(i)) of 128 bits of the sequence are combined with successive 64-bit blocks of plaintext (P (i)) to produce successive 64-bit blocks of ciphertext. The blockwise use of a long pseudo-random sequence preserves the advantages of a block cipher in terms of data confidentiality and data integrity, as well as benefiting from the speed advantages of a stream cipher.
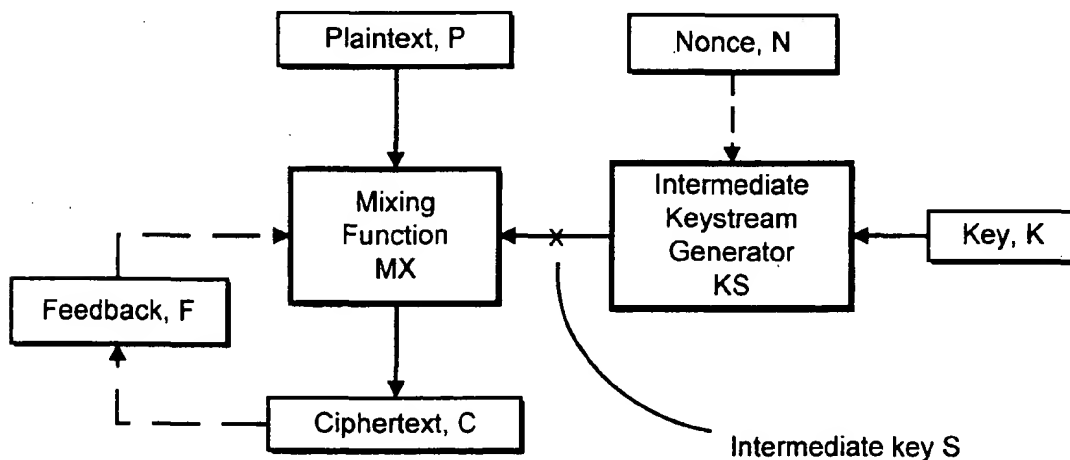
Figure 2

EP 1 063 811 A1

Description

[0001]   The present invention relates to an encryption apparatus and method, and to a corresponding decryption apparatus and method, particularly but not exclusively for use in telecommunication systems and over computer networks, for example to provide secure channels for Internet commerce.

[0002]   Modern cryptography. relies on the use of cryptographic algorithms, also known as ciphers, to encrypt data for secure transmission over networks such as the Internet. In addition to the primary. function of preserving data confidentiality, cryptographic systems can serve a number of other functions, including preserving data integrity. This requires that the receiver of a message should be able to verify that the message has not been changed by a third party after encryption.

[0003]   To avoid the need to keep cryptographic algorithms secret, which can be difficult when the algorithm is incorporated into commercially available products, present day algorithms use an encryption key K, which can assume any one of a large number of possible values. The security of the encryption system then resides in the key, rather than in the details of the algorithm.

[0004]   There are two general types of key-based algorithm, known as symmetric, or private-key, algorithms and asymmetric, or public-key algorithms. The present invention is concerned with symmetric algorithms. In most symmetric algorithm based encryption schemes, the encryption and decryption keys are the same and the sender and receiver must agree on a key before they can communicate securely. They must also ensure that the key remains secret, since the security of the system is compromised by a third party knowing the key.

[0005]   Figure 1 illustrates the basic principle of encrypted communication. Unencrypted data, known as plaintext P, is encrypted by the Sender using an encryption algorithm E and an encryption key K to produce encrypted data, known as ciphertext C. Mathematically, the operation can be written as $E_K(P) = C$. The ciphertext is sent over a communication channel to a Receiver, where a decryption algorithm D is applied to recover the plaintext from the ciphertext. This operation can be written as $D_K(C) = P$. The functions D and E have the property that $D_K(E_K(P)) = P$, and D is referred to herein as the inverse of E. In the event that different, but cryptographically related, keys K_1, K_2 are used for encryption and decryption respectively, then the appropriate representations are $E_{k1}(P) = C$, $D_{K2}(C) = P$ and $D_{K2}(E_{k1}(P)) = P$.

[0006]   Symmetric algorithms can be divided into two categories, generally referred to as block ciphers and stream ciphers. In general terms, stream ciphers operate on plaintext a single bit at a time and are considerably faster than block ciphers. For example, in a simple stream cipher, each bit of plaintext is exclusive or'd (XORed) with a respective bit from a pseudo-random stream of bits generated by a keystream generator.

[0007]   Although a stream cipher provides the primary cryptographic function of preserving the confidentiality of a message, the preservation of data integrity requires an additional independent mechanism, since otherwise changes at one bit position in the plaintext result in changes at the corresponding position in the ciphertext. Therefore, additional circuits and/or memory may be required to implement an independent integrity preserving mechanism.

[0008]   Block ciphers operate on groups of plaintext bits known as blocks. For example, a typical block size is 64 bits. Block ciphers are used because they are capable of providing both confidentiality and data integrity. However, they are in general considerably slower than stream ciphers. For example, the known DES block cipher encrypts data in 64-bit blocks, using a 56-bit key. This is used to generate a key schedule comprising 16 different 48-bit subsets of the key. All the outputs of the key schedule are used for the encryption of every plaintext block in a highly non-linear data randomising algorithm.

[0009]   Reference is directed to Bruce Schneier, "Applied Cryptography", Second Edition, Chapter 9, pp. 189 - 211, for an overview of block and stream ciphers and their modes of operation.

[0010]   The present invention aims to combine the advantages of block and stream ciphers.

[0011]   In accordance with the present invention, there is provided encryption apparatus for performing symmetric encryption, comprising means for generating a keystream block in response to an encryption key, the keystream block comprising a plurality of bits of a pseudo-random number sequence, and means for combining the keystream block with a plaintext block comprising a plurality of bits to produce a ciphertext block, wherein the number of bits in the keystream block exceeds the number of bits in the plaintext block.

[0012]   The blockwise use of a pseudo-random sequence generated from a shorter key sequence can preserve the advantages of a block cipher in terms of data confidentiality and data integrity, as well as enabling the apparatus to benefit from the speed advantages of a stream cipher, since generation of a pseudo-random number sequence is generally faster than operation of a block cipher and can allow simpler and therefore faster implementation of the combining means.

[0013]   The generating means can comprise means for selecting successive keystream blocks from said pseudo-random number sequence, each to be combined with a respective plaintext block from plaintext data to be encrypted. Further, the combining means can be configured to use each of the successive keystream blocks once only during the encryption of given plaintext data under a given key, which can provide unconditional, one-time-pad-like, security if the pseudo-random number generator is assumed to be secure. Differently stated, the generating means is capable of

generating fresh keystream blocks to encrypt every plaintext block, with the length of the keystream bit sequence being dependent on the length of the data to be encrypted. The freshness of the keystream data enables the combining means to be simpler and faster than the corresponding data randomisation means in a block cipher, since the combining means is not required to be a non-linear function.

[0014] The apparatus can include means for splitting a keystream block into sub-blocks, which can include distinct or overlapping portions of the pseudo-random bit sequence.

[0015] The combining means can comprise first means for combining a first portion of the keystream block with the plaintext block to produce a first combination, and second means for combining the result of said first combination with a second part of the keystream block to produce the ciphertext block. By splitting an s-bit keystream block into b-bit sub-blocks, which may involve the addition of padding bits, the combining means can combine equal length sequences of plaintext and keystream sub-blocks, for example by mathematical operations in $GF(2^b)$.

[0016] The generating means can comprise a pseudo-random number generator having an encryption key as an input, or a hash function receiving a key and nonce as inputs.

[0017] The encryption apparatus can comprise a plurality of keystream generators for generating intermediate keys and means for switching between said keystream generators in dependence on the plaintext block to be encrypted, so allowing for increases in speed by parallel computation of intermediate key sequences.

[0018] The encryption apparatus can alternatively comprise a second keystream generator for generating a second intermediate key in response to the first intermediate key generated by the first keystream generator with a nonce input. By creating nonces based on individual elements such as files and records, the first intermediate key provides individual element encryption, which provides for encryption of individual elements in applications such as database and disk encryption, and permits the secret key to be maintained at the encryption site, with sharing of the intermediate key only.

[0019] According to the invention, there is further provided encryption apparatus comprising a keystream generator for generating a pseudo-random number sequence in response to an encryption key and processing circuitry configured to combine a keystream block comprising a bit pattern selected from the pseudo-random number sequence with a plaintext block comprising a bit sequence, said keystream block bit pattern comprising a greater number of bits than the plaintext block bit sequence.

[0020] In accordance with the invention, there is also provided decryption apparatus for performing symmetric decryption, comprising means for generating a keystream block in response to a decryption key, the keystream block comprising a plurality of bits of a pseudo-random number sequence, and means for combining the keystream block with a ciphertext block comprising a plurality of bits to recover a plaintext block, wherein the number of bits in the keystream block exceeds the number of bits in the ciphertext block.

[0021] The invention further provides a method of performing symmetric encryption comprising generating a keystream block in response to an encryption key, the keystream block comprising a plurality of bits of a pseudo-random number sequence, and combining the keystream block with a plaintext block comprising a plurality of bits to produce a ciphertext block, wherein the number of bits in the keystream block exceeds the number of bits in the plaintext block.

[0022] The invention further provides a method of decrypting a symmetrically encrypted ciphertext message comprising generating a keystream block in response to a decryption key, the keystream block comprising a plurality of bits of a pseudo-random number sequence, and combining the keystream block with a ciphertext block comprising a plurality of bits to produce a plaintext block, wherein the number of bits in the keystream block exceeds the number of bits in the ciphertext block.

[0023] In accordance with the invention, there is further provided a computer program for performing symmetric encryption comprising means for generating a keystream block in response to an encryption key, the keystream block comprising a plurality of bits of a pseudo-random number sequence, and means for combining the keystream block with a plaintext block comprising a plurality of bits to produce a ciphertext block, wherein the number of bits in the keystream block exceeds the number of bits in the plaintext block.

[0024] There is also provided, in accordance with the invention, a computer program for decrypting a symmetrically encrypted ciphertext message comprising means for generating a keystream block in response to a decryption key, the keystream block comprising a plurality of bits of a pseudo-random number sequence, and means for combining the keystream block with a ciphertext block comprising a plurality of bits to produce a plaintext block, wherein the number of bits in the keystream block exceeds the number of bits in the ciphertext block.

[0025] Embodiments of the invention will now be described by way of example with reference to the accompanying drawings, in which:

Figure 1 is a schematic block diagram showing the known structure of a symmetric cryptographic system;

Figure 2 is a schematic block diagram showing the basic structure of the encryption apparatus in accordance with the invention;

Figure 3 is a schematic block diagram showing one implementation of the keystream generator KS shown in Figure

2;

Figure 4 is a schematic block diagram showing a further implementation of the keystream generator KS shown in Figure 2;

Figure 5 is a schematic block diagram showing a basic structure of decryption apparatus according to the invention;

Figure 6 is a block diagram illustrating a message format for verifying data integrity;

Figure 7 is a flow diagram explaining the operation of the encryption apparatus of Figure 2;

Figure 8 illustrates the generation of blocks of ciphertext in accordance with the invention;

Figure 9 is a flow diagram explaining the operation of the decryption apparatus of Figure 5;

Figure 10 is a schematic block diagram showing an example of a mixing function MX;

Figure 11 is a block diagram showing a more detailed implementation of the mixing function MX shown in Figure 10;

Figures 12-19 are schematic block diagrams showing various implementations of the mixing function MX with feedback;

Figure 20 is a schematic block diagram showing an example of an inverse mixing function MX^{-1} corresponding to the mixing function MX shown in Figure 10;

Figure 21 is a block diagram showing a more detailed implementation of the inverse mixing function MX^{-1} shown in Figure 20;

Figure 22 is a block diagram showing an encryption apparatus comprising a counter as a nonce input;

Figure 23 is a block diagram showing a model for parallel computation;

Figure 24 is a block diagram illustrating an extension of the KS function;

Figure 25 illustrates an application of the apparatus of Figure 24 as a disk encryption system;

Figure 26 illustrates an example of encryption with a nonce, applied to a communications system;

Figure 27 illustrates a software implementation of the encryption scheme of Figure 26;

Figure 28 shows a hardware implementation of the encryption scheme of Figure 26;

Figure 29 illustrates an example of an encryption system used in broadcasting;

Figure 30 is a schematic block diagram illustrating a method of enhancing key confidentiality by limiting key exchange using nonces;

Figure 31 illustrates an example of a disk encryption system;

Figure 32 illustrates an example of a database encryption system;

Figure 33 shows an example of a multimedia distribution system;

Figure 34 is a schematic diagram of the encryption software for the multimedia system of Figure 33; and

Figure 35 is a schematic diagram of the decryption software for a consumer device to be used with the multimedia distribution system of Figure 33.

[0026] Referring to Figure 2, the encryption apparatus according to the invention comprises an intermediate keystream generator KS and a mixing function MX.

Keystream generator KS

[0027] KS is designed to generate an intermediate key, in the form of a pseudo-random number sequence S, in response to an encryption key K. KS is, for example, a known cryptographic pseudo-random number generator or an ideal cryptographic hash function or any device which is guaranteed to generate the same unpredictable sequence whenever the same input is given. For example, KS is a pseudo-random number generator used for known stream ciphers, such as WAKE, SEAL or RC4. KS can also be a block cipher in counter mode or a hash function with a counter. While a pseudo-random number generator can generate a very long plaintext sequence, for example 64kB with the SEAL algorithm, a hash function generates a relatively short sequence, for example 160 bits. However, by using different successive inputs to the hash function, as described in detail below, the hash function operates as pseudo-random number generator to generate a long pseudo-random sequence.

[0028] As shown in Figure 2, KS can also receive a nonce N, defined as a sequence of data in which, for a particular key, there is a negligible probability that a particular value will appear twice. For example, a nonce can be derived from a counter which always counts up during the encryption of a single message under a particular key. Therefore, each of the nonce values used during the encryption of the message is unique. Further examples of nonces are given below.

[0029] As a further example of an implementation of KS, referring to Figure 3, a nonce N is encrypted using a known block cipher under a key K to produce ciphertext S which comprises a pseudo-random number string. Alternatively, referring to Figure 4, a concatenation of the key K, a nonce N and padding bits is used as an input to a hash function to generate a hash value to be used as the intermediate key S. In both of the above cases, another nonce, for example the previous nonce value incremented by one, is used to generate the next pseudo-random number string, once the current string has been used for encryption.

Mixing function MX

[0030]   Referring again to Figure 2, the intermediate key S is an input to the mixing function MX which receives a plaintext message P and combines it with the pseudo-random number sequence S to generate the ciphertext C. The mixing function MX can also receive a feedback input F, dependent on previous ciphertext output.

[0031]   Referring to Figure 5, the mixing function MX is a reversible function, so that decryption is achieved by applying the ciphertext C and the pseudo-random sequence S to the inverse mixing function MX^{-1} to restore the plaintext P.

[0032]   The mixing function MX can be designed to preserve data confidentiality and data integrity in accordance with the following criteria:

Confidentiality - the ciphertext of a plaintext can be all possible strings with substantially uniform distribution under any key.

Integrity - the ciphertext of a modified plaintext is substantially uniformly distributed amongst all possible ciphertext strings under any key, so that an attacker who does not know S cannot predict how the plaintext differs after any modification of the ciphertext.

[0033]   These conditions can be represented mathematically as follows:

Confidentiality - $Prob_s(MX(P,S)=C)<e_c$, for any P and C

Integrity - $Prob_s(MX(P,S) \wedge MX(P',S)=D)<e_i$, for any (P,P',D)

where $Prob_y(X)$ represents the probability of X over the uniform distribution of Y, the operator '$\wedge$' represents bitwise exclusive-or (XOR) and where e_c and e_i are sufficiently small that it is computationally infeasible for an attacker to exploit these probabilities to crack the encryption scheme.

[0034]   Referring to Figure 6, verification of data integrity can be performed by attaching known redundant data to a plaintext message and confirming that, after encryption and corresponding decryption, the known data is recovered.

[0035]   Figure 7 is a flow diagram explaining the operation of the apparatus of Figure 2, without nonce or feedback inputs, by reference to a generalised block P(i), of length b bits, which represents the ith block of the plaintext message P. C(i) represents the corresponding block of ciphertext, of length b bits. S(i) represents the ith block of the pseudo random number sequence S, of length s bits.

[0036]   At step s1, KS generates the pseudo-random number sequence S corresponding to the key K, where the total length of the sequence S is generally greater than the length of the key K (k bits). At step s2, the ith block of S is combined with the ith block of the plaintext P by the mixing algorithm MX to produce the ith block of ciphertext C. This step is repeated for all n blocks of P. For example, referring to Figures 7 and 8 and assuming the total length of the plaintext data is 6400 bits, and that b = 64, k = 64 and s = 2b = 128, then n = 100 and i takes the values 1 to 100. The sequence S must then be at least 12800 bits long. Step s2 is repeated for each value of i, with the first step being the combination of P(1), ie. the first 64 bits of P, with S(1), ie. the first 128 bits of S, to produce C(1) ie. the first 64 bit block of ciphertext C.

[0037]   By using an s-bit intermediate key which is longer than a b-bit plaintext message, information theory explains that perfect data confidentiality can be achieved. In terms of data integrity, there are, on average, 2^{s-b} key candidates for each plaintext/ciphertext pair, so that even if a cryptanalyst knows both the plaintext and the ciphertext, he cannot predict how the plaintext will change for any amendment to the ciphertext, assuming that the probability of doing so, $1/2^{s-b}$, is negligible. For example, where s=2b and b = 64 bits, then the number of key candidates is 2^{64}.

[0038]   Figure 9 is a flow diagram explaining the operation of the decryption apparatus of Figure 5, and is analogous to the flow diagram of Figure 7.

[0039]   At step s3, KS generates the pseudo-random number sequence S corresponding to the key K, of length k bits. At step s4, the ith block of S is combined with the ith block of the ciphertext C by the inverse mixing algorithm MX^{-1} to produce the ith block of plaintext P. This step is repeated for all n blocks of C to recover the original plaintext.

[0040]   Referring to Figure 10, one example of an MX function comprises two operator functions, each of which operate on separate parts of the pseudo-random input block S(i). For example, a block of plaintext P(i) comprises b bits. S(i) is a 2b bit block. The first b bits of S(i) are denoted A(i) and the final b bits are denoted B(i), so that S(i) can also be represented as a concatenation of A(i) and B(i), i.e. A(i)|| B(i). A(i) is combined with the b-bit block P(i) by multiplication in $GF(2^b)$ and the result is combined with B(i) by addition in $GF(2^b)$, where multiplication and addition in $GF(2^b)$ refers to computation in a Galois Field. Such computation can readily be implemented in hardware with linear-feedback shift registers. Reference is directed to Bruce Schneier, "Applied Cryptography", Second Edition, pages 254 to 255 for further details of Galois Field computation.

[0041]   For multiplication and division in $GF(2^b)$ a modification needs to be applied to each pseudo-random sequence block whose value is 0, to ensure that the function is reversible. The problem can be illustrated with the case of integer

multiplication, where if a*c=b, then b/c=a except for c=0. However, by modifying the multiplier c to avoid zero values, this problem can be overcome.

[0042] The necessary modification can be carried out in a number of known ways. For example, the block A(i) is tested. If it is 0, it is reset to some known number, for example 1. Alternatively, the entire block is discarded and the next block chosen. This procedure can be carried out at the key sequence generator KS. For example, if a key sequence block S(i) is to be split into a plurality of sub-blocks T(i) ... Z(i) and it is known that each sub-block is to be processed at the mixing function MX using multiplication/division in GF(2^b), then each sub-block is checked and if its value is 0, is reset to be 1. Alternatively, if any of the sub-blocks T(i) ... Z(i) have a value of 0, the entire block S(i) is discarded and the next block selected from the pseudo-random number sequence from the pseudo-random number generator.

[0043] The MX function shown in Figure 10 can also be represented as:

$$C(i) = MX(P(i), S(i)) = (A(i)^*P(i)) \wedge B(i)$$

where

$$S(i) = A(i)||B(i);$$

$X \wedge Y$ represents addition in GF(2^b), equivalent to bitwise exclusive-or (XOR); and
X^*Y represents multiplication in GF(2^b);

[0044] Referring to Figure 11, a more detailed implementation of the encryption function of Figure 10 is shown. A 64-bit key K is used as an input to the pseudo-random number generator used for the WAKE stream cipher, which produces a stream of 32-bit words. Four such words are stored in a 128-bit buffer BUF128 to provide the keystream block S(i). The 128-bit block S(i) is split into two 64-bit blocks A(i) and B(i). Plaintext P is stored 64 bits at a time in a 64-bit plaintext buffer PBUF. The MX function relies on multiplication and addition in GF (2^{64}) as explained with reference to Figure 10.

[0045] The resulting ciphertext block C(i) is stored 64 bits at a time in a 64-bit ciphertext buffer CBUF and the whole ciphertext is generated by concatenation of the resulting ciphertext blocks, i.e. C(1)||C(2)||C(3) ... || C(n), where n is the total number of plaintext blocks. Reference is directed to Bruce Schneier, "Applied Cryptography", Second Edition, pages 400 to 402 for a more detailed description of, and references to, the WAKE algorithm.

[0046] Another example of a mixing function suitable for use in the encryption devices of Figures 10 and 11 and using the terminology set out above can be represented as:

$$MX(P(i), S(i)) = A(i) \wedge (P(i)^*B(i))$$

[0047] The block length b can be varied dynamically, as long as both encryption and decryption devices have access to the necessary common information, in particular the block length and the definition of the irreducible polynomial for GF(2^b). However, memory consumption and performance constraints may limit the use of this technique for large sized blocks, for example in excess of 1MB.

[0048] The keystream block S(i) need not be split into distinct sub-blocks, but can also be split into sub-blocks containing overlapping bit sequences. For example, a 128-bit keystream block S(i) is split into 16 sub-blocks of 48 bits each. Each sub-block can then be used as an input to one round of the DES algorithm to encrypt a plaintext block. Reference is directed to Bruce Schneier, "Applied Cryptography", Second Edition, pages 270-278 for a description of the DES algorithm.

[0049] Examples of the mixing function MX with a feedback input are shown in Figures 12 to 15, using the terminology set out above. As described above, each block S(i) of the pseudo-random sequence S is divided into two b-bit blocks so that S(i) = A(i)||B(i), where A(i) and B(i) are each b bits long. Each block of ciphertext C(i) is a combination of a plaintext block S(i) and the previous ciphertext block C(i-1), represented as C(i) = MX(P(i), S(i), C(i-1)). In all cases, C(0) is an initial vector IV supplied to prevent encryption attacks based on messages having identical beginnings, for example, a standard e-mail address. IV can be random data or, for example, a time stamp.

[0050] Figure 12 represents the mixing function:

$$C(i) = MX(P(i), S(i), C(i-1)) = ((C(i-1)^*A(i)) \wedge P(i))^*B(i)$$

6

[0051] Figure 13 represents the mixing function:

$$C(i) = MX(P(i), S(i), C(i-1)) = ((C(i-1) \wedge A(i))^* P(i)) \wedge B(i)$$

[0052] Figure 14 represents the mixing function:

$$C(i) = MX(P(i), S(i), C(i-1)) = ((C(i-1)^* A(i)) \wedge (P(i)^* B(i)))$$

[0053] Figure 15 represents the mixing function:

$$C(i) = MX(P(i), S(i), C(i-1)) = (C(i-1) \wedge A(i))^* (P(i) \wedge B(i)))$$

[0054] S(i) can be divided into more than two blocks. For example, referring to Figures 16 to 19, a pseudo-random sequence S(i) of length 3b bits, is divided into three blocks A(i), B(i) and D(i), each of length b bits.
[0055] Figure 16 represents the mixing function:

$$C(i) = MX(P(i), S(i), C(i-1)) = (((C(i-1)^* D(i)) \wedge P(i))^* A(i)) \wedge B(i)$$

[0056] Figure 17 represents the mixing function:

$$C(i) = MX(P(i), S(i), C(i-1)) = (((C(i-1) \wedge D(i))^* P(i)) \wedge A(i))^* B(i)$$

[0057] Figure 18 represents the mixing function:

$$C(i) = MX(P(i), S(i), C(i-1)) = ((C(i-1)^* D(i)) \wedge (P(i)^* A(i)))^* B(i)$$

[0058] Figure 19 represents the mixing function:

$$C(i) = MX(P(i), S(i), C(i-1)) = ((C(i-1) \wedge D(i))^* P(i) \wedge A(i))) \wedge B(i)$$

[0059] Referring to Figure 20, an inverse function (MX^{-1}) of the MX function shown in Figure 10 comprises two operator functions, each of which operate on separate parts A(i) and B(i) of the pseudo-random input block S(i). For example, B(i) is combined with the b-bit block C(i) by addition in $GF(2^b)$ and the result is combined with A(i) by division in $GF(2^b)$, where division and addition in $GF(2^b)$ refers to computation in a Galois Field as explained in relation to Figure 10.
[0060] Referring to Figure 21, a more detailed implementation of the decryption function of Figure 20 is shown. A 64-bit key K is used as an input to the WAKE stream cipher, which produces a stream of 32-bit words. Four such words are stored in a 128-bit buffer BUF128 to provide the keystream block S(i). The 128-bit block S(i) is split into two 64-bit blocks A(i) and B(i). Ciphertext C is stored 64 bits at a time in a 64-bit ciphertext buffer CBUF. The MX^{-1} function relies on division and addition in $GF(2^{64})$ as explained with reference to Figure 20.
[0061] The resulting plaintext block P(i) is stored 64 bits at a time in a 64-bit plaintext buffer PBUF and the whole plaintext is generated by concatenation of the resulting plaintext blocks, i.e. P(1)II(P(2)IIP(3)...IIP(n), where n is the total number of plaintext blocks.

Encryption with Nonce

[0062] As mentioned above, if an encryption scheme uses a nonce as an input value, the nonce is designed so as to avoid generation of the same key value. For example, referring to Figure 22, the encryption apparatus includes a nonce input to the key sequence generator KS, where the nonce values are generated by a counter CT 1 which counts from 1 to n, where n is the number of plaintext blocks to be encrypted. Therefore, the nonce has a unique value for each plaintext block to be encrypted.

7

[0063]   Advantageously, if the nonce sequence is readily predictable, parallel computation is facilitated. For example, referring to Figure 23, parallel computation is implemented by two independent keystream generators KS 1 and KS2. Each counter CT1, CT2 is synchronised to the block number i but produces a single increment every two blocks, with one counter CT1 causing KS1 to produce the block sequence S(1), S(3), S(5) ... S(n-1), and the other counter CT2 causing KS2 to produce the block sequence S(2), S(4), S(6), ... S(n). A switching element SW switches between the outputs of KS1 and KS2 so that each plaintext block P(i) is mixed by the function MX with the correct block S(i) to produce the ciphertext block C(i). The speed of the switching function is greater than that of the intermediate key generating step, so that an overall increase in operating speed results. This scheme can be generalised to realise parallel computation by a factor r, using a plurality of keystream generators, KSi, where i = 1, ..., r, each of which generates only $S(((j-1) \times r) + i)$, where j = 1,... , n/r, where 'x' represents integer multiplication.

[0064]   Referring to Figure 24, the pseudo-random sequence S1 generated using a nonce input N1 to a keystream generator KS1 can be used as the input to a second keystream generator KS2, which receives a second nonce input N2 to produce a second sequence S2. Therefore, two intermediate keys S1, S2 are produced. The arrangement of Figure 24 can be used as a disk encryption system, as shown in Figure 25. KS1 generates a file dependent key S1 out of the secret key K and the nonce input N1, which comprises a concatenation of the file name with a time stamp and the identity of the file owner. KS2 receives the key S1 and generates the intermediate key S2 which is used by the function MX to encrypt the original file data. This provides increased security over a disk encrypted with a single key, since the encryption key S1 differs from file to file, and the master key K is kept secret at the encryption site.

[0065]   Various examples of the application of the encryption and decryption apparatus and techniques according to the invention will now be described with reference to Figures 26 to 33.

[0066]   Referring to Figure 26, in a communications system, a receiver and a transmitter share a key and synchronise counters CT1, CT2, so that KS generates the same value at the receiver and at the transmitter. Consequently, each encrypted plaintext block C(i) can be decrypted by operating on it with the inverse mixing function MX^{-1} to recover the original plaintext block P(i).

[0067]   Referring to Figure 27, the encryption scheme for the communications system of Figure 26 can be implemented in software. The sender and receiver exchange keys and the synchronising counter value using a known key exchange protocol, for example Otway-Rees, Kerberos or, if public-key cryptography is used, Diffie-Hellman. Reference is further directed to Bruce Schneier, "Applied Cryptography", Second Edition, pp. 59-60 and 513 - 515. Encryption program code for implementing the KS and MX functions shown in Figure 26 and previously described is stored in a memory SMEM. A CPU at the Sender side executes the encryption code to encrypt a message. The encrypted message is transferred over the network INET via the sender's I/O device, received by the receiver's I/O device, and decrypted by the receiver CPU running the decryption code, which implements KS and MX^{-1} as previously described, which is stored in memory RMEM, to recover the message. The software implementation is suitable for use in a wide variety of devices including devices such as PCs, smart cards, network routers and in remote controllers for equipment such as car security devices.

[0068]   Referring to Figure 28, encryption and decryption can also be implemented as dedicated hardware circuits, for example embedded in the I/O devices of sender and receiver respectively. The encryption circuit E implements the KS and MX functions, while the decryption circuit D implements the KS and MX^{-1} functions, both as previously described above. The Sender's CPU sends a plaintext message to the sender's I/O device, so that the message is encrypted by the encryption circuit E using a key and counter value which is shared with the receiver prior to the communication using a key exchange protocol. The encrypted message is sent via the network INET and is received by the receiver's I/O device. The embedded decryption circuit D decrypts the message which is then stored in memory by the receiver's CPU. Dedicated hardware encryption/decryption circuits are used in devices such as high-end smart cards and smart card readers, mobile terminals, PCs, TV, DVD players and so on.

[0069]   Referring to Figure 29, in a broadcasting system, a broadcast station BSTN generates a programme key, and a nonce which comprises a programme identifier and frame number of the contents data where a programme is divided into several frames. The contents data is encrypted using the key and nonce. A key exchange protocol is used to transmit the key to a consumer device, for example a digital TV, which contains the inverse decryption function MX^{-1}. Knowing the key and the programme identifier and frame numbers of the programme which the consumer wishes to view, the consumer device can decrypt the programme. The key server at the broadcast station can generate intermediate keys based on the frame information, so that intermediate frame keys are sent to the consumer device during a programme. To receive the whole programme, a third party would then have to cryptanalyse all of the frame keys.

[0070]   Referring to Figure 30, a nonce can be used to enhance key confidentiality. The first key exchange between two devices must exchange both the key and the nonce. However, when the key value is to be updated, for example, for the next frame of data, only the nonce need be exchanged. The secret key need therefore only be exchanged once, whereas the nonce can be repeatedly exchanged, without compromising the security of the system, if it becomes known. If the system has a way of nonce-updating, for example a counter to track frame numbers, then the first nonce need only be used for frame synchronisation and further nonces need not be sent.

[0071]   Referring to Figure 31, in an implementation of a disk encryption system as described with reference to Figure 25, files are encrypted under different keys, where each nonce comprises a concatenation of a file name with a time stamp and an owner identity. The CPU implements the encryption algorithm stored in memory DMEM, and described in relation to Figure 25. The secret key is kept on a smart card by the operator of the encryption system.

[0072]   Referring to Figure 32, which illustrates a further application of the encryption algorithm described with reference to Figure 25 for a database encryption system, the nonce comprises a concatenation of ID, category and revision number or time stamp.

[0073]   Figure 33 illustrates a further application of the encryption algorithm described with reference to Figure 25 for use in a multimedia distribution system. Content on a CD- or DVD- ROM is encrypted using a secret key K and a nonce N1 which comprises, for example, a concatenation of the program name with the date and a version number. For example, referring to Figure 34, a number of programmes are encrypted onto DVD-ROM, each programme being encrypted under its own key S1 using the master key K and the nonce N1, which is unique to each film on the DVD-ROM. Copies of the DVD-ROMs are distributed to consumers free of charge. When a consumer wishes to play one of the films on the encrypted disk, the consumer contacts key server KS1 with his requirements and makes the appropriate payment, for example by credit card over the Internet. The key server KS1 calculates the key S1 to be sent to the consumer from the master key K and the programme titles and other data, stored at the key server KS1, which make up nonce N1. A key exchange protocol is then initiated and the consumer device receives the appropriate intermediate key S1. Referring to Figure 35, the consumer device uses the key S1 as the input to a keystream generator KS2 to generate the correct intermediate key S2. Therefore, referring again to Figure 33, the required film can be decrypted by the consumer's CPU running the decryption program code stored in memory CMEM, which implements the functionality shown in Figure 35.

Claims

1.   Encryption apparatus for performing symmetric encryption, comprising:

   means (KS) for generating a keystream block (S(i)) in response to an encryption key (K), the keystream block comprising a plurality of bits (s) of a pseudo-random number sequence; and
   means (MX) for combining the keystream block (S(i)) with a plaintext block (P(i)) comprising a plurality of bits (b) to produce a ciphertext block (C(i));

   wherein the number of bits (s) in the keystream block exceeds the number of bits (b) in the plaintext block.

2.   Apparatus according to claim 1, wherein the generating means comprises means for selecting successive keystream blocks (S(i)) from said pseudo-random number sequence, each to be combined with a respective plaintext block (P(i)) from plaintext data (P) to be encrypted.

3.   Apparatus according to claim 2, wherein the combining means is configured to use each of the successive keystream blocks once only during the encryption of plaintext data under a given key.

4.   Apparatus according to claim 2 or 3, wherein the selecting means is configured to select non-overlapping bit sequences from said pseudo-random number sequence to form each of the keystream blocks.

5.   Apparatus according to any one of the preceding claims, wherein the or each keystream block S(i) comprises at least twice as many bits (2b, 3b) as the or each plaintext block P(i).

6.   Apparatus according to any one of the preceding claims, further comprising means for splitting the or each keystream block (S(i)) into sub-blocks (A(i), B(i), D(i)).

7.   Apparatus according to claim 6, wherein the sub-blocks have the same number of bits as the plaintext block (P(i)) to be encrypted.

8.   Apparatus according to any one of the preceding claims, wherein the combining means comprises first means for combining a first portion (A(i)) of the keystream block (S(i)) with the plaintext block (P(i)) to produce a first combination, and second means for combining the result of said first combination with a second part (B(i)) of the keystream block S(i) to produce the ciphertext block (C(i)).

9. Apparatus according to claim 8, wherein said first and second combining means each comprise different mathematical operations.

10. Apparatus according to claim 8 or 9, wherein one of the first and second combining means comprises means for multiplying in $GF(2^b)$.

11. Apparatus according to any one of claims 8 to 10, wherein one of the first and second combining means comprises means for adding in $GF(2^b)$.

12. Apparatus according to any one of claims 8 to 11, further comprising a third combining means for combining a third portion $(D(i))$ of the keystream block $S(i)$ with a ciphertext block $(C(i-1))$ to provide a feedback input.

13. Apparatus according to any one of the preceding claims, wherein said generating means further comprises a nonce input.

14. Apparatus according to claim 13, comprising a counter (CT1, CT2) for generating the nonce input.

15. Apparatus according to claim 13 or 14, wherein said generating means comprises a hash function generator.

16. Apparatus according to any one of the preceding claims, comprising a plurality of keystream generators (KSi) for generating intermediate keys and means (SW) for switching between said keystream generators in dependence on the plaintext block $(P(i))$ to be encrypted.

17. Apparatus according to any one of claims 1 to 15, wherein the keystream generator comprises a first keystream generator for generating a first intermediate key (K1), further comprising a second keystream generator for generating a second intermediate key (K2) in response to the first intermediate key.

18. Apparatus according to claim 17, wherein the first intermediate key comprises an encryption key for an individual element selected from a plurality of elements to be encrypted.

19. Encryption apparatus comprising

a keystream generator (KS) for generating a pseudo-random number sequence in response to an encryption key (K); and

processing circuitry (MX) configured to combine a keystream block $(S(i))$ comprising a bit pattern selected from the pseudo-random number sequence with a plaintext block $(P(i))$ comprising a bit sequence, said keystream block bit pattern comprising a greater number of bits than the plaintext block bit sequence.

20. Decryption apparatus for performing symmetric decryption, comprising:

means (KS) for generating a keystream block $(S(i))$ in response to a decryption key (K), the keystream block comprising a plurality of bits (s) of a pseudo-random number sequence; and

means (MX^{-1}) for combining the keystream block $(S(i))$ with a ciphertext block $(C(i))$ comprising a plurality of bits (b) to recover a plaintext block $(P(i))$;

wherein the number of bits (s) in the keystream block exceeds the number of bits (b) in the ciphertext block.

21. Apparatus according to claim 20, comprising generating means as claimed in any one of claims 1 to 19.

22. Apparatus according to claim 20 or 21, wherein the combining means is the inverse of the combining means claimed in any one of claims 1 to 19.

23. Decryption apparatus configured to decrypt ciphertext encrypted by the apparatus of any one of claims 1 to 19.

24. A method of performing symmetric encryption comprising:

generating a keystream block $(S(i))$ in response to an encryption key (K), the keystream block comprising a plurality of bits (s) of a pseudo-random number sequence; and

10

combining the keystream block (S(i)) with a plaintext block (P(i)) comprising a plurality of bits (b) to produce a ciphertext block (C(i));

wherein the number of bits (s) in the keystream block exceeds the number of bits (b) in the plaintext block.

25. A method according to claim 24, comprising the step of selecting successive key sequence blocks from the pseudo-random number sequence, each to be combined with a respective plaintext block from a plaintext message (P).

26. A method according to claim 25, comprising using each of the successive keystream blocks once only during the encryption of a given plaintext message to encrypt a single plaintext block P(i).

27. A method of decrypting a symmetrically encrypted ciphertext message (C) comprising:

generating a keystream block (S(i)) in response to a decryption key (K), the keystream block comprising a plurality of bits (s) of a pseudo-random number sequence; and
combining the keystream block (S(i)) with a ciphertext block (C(i)) comprising a plurality of bits (b) to produce a plaintext block (P(i));

wherein the number of bits (s) in the keystream block exceeds the number of bits (b) in the ciphertext block.

28. A computer program for performing symmetric encryption comprising

means for generating a keystream block (S(i)) in response to an encryption key (K), the keystream block comprising a plurality of bits (s) of a pseudo-random number sequence; and
means for combining the keystream block (S(i)) with a plaintext block (P(i)) comprising a plurality of bits (b) to produce a ciphertext block (C(i));

wherein the number of bits (s) in the keystream block exceeds the number of bits (b) in the plaintext block.

29. A computer program for decrypting a symmetrically encrypted ciphertext message (C) comprising:

means for generating a keystream block (S(i)) in response to a decryption key (K), the keystream block comprising a plurality of bits (s) of a pseudo-random number sequence; and
means for combining the keystream block (S(i)) with a ciphertext block (C(i)) comprising a plurality of bits (b) to produce a plaintext block (P(i));

wherein the number of bits (s) in the keystream block exceeds the number of bits (b) in the ciphertext block.

30. A computer readable medium carrying a computer program according to claim 28 or 29.

11

02/11/2004, EAST Version: 1.4.1

Figure 1

Figure 2

Key, K

Nonce, N → Block Cipher → Intermediate key S

Figure 3

Padding	Key, K	Padding	Nonce, N	Padding

Hash Function → Intermediate key S
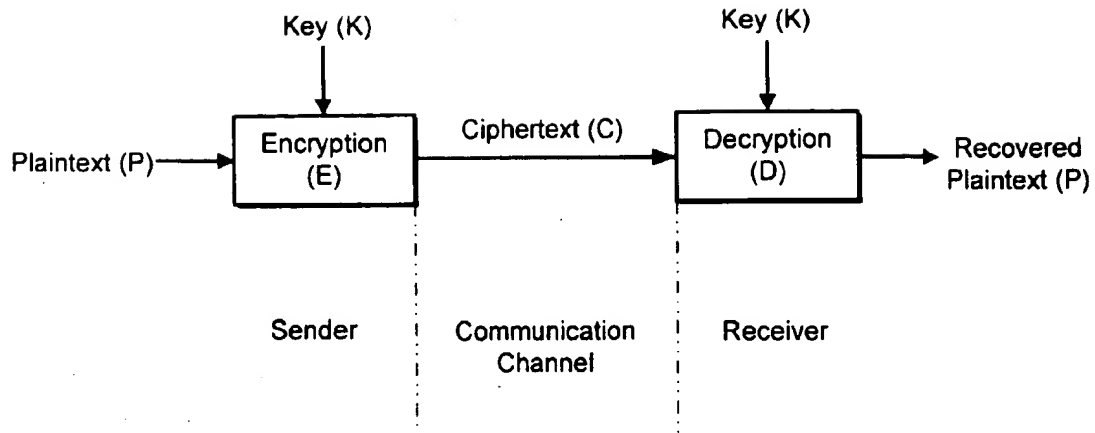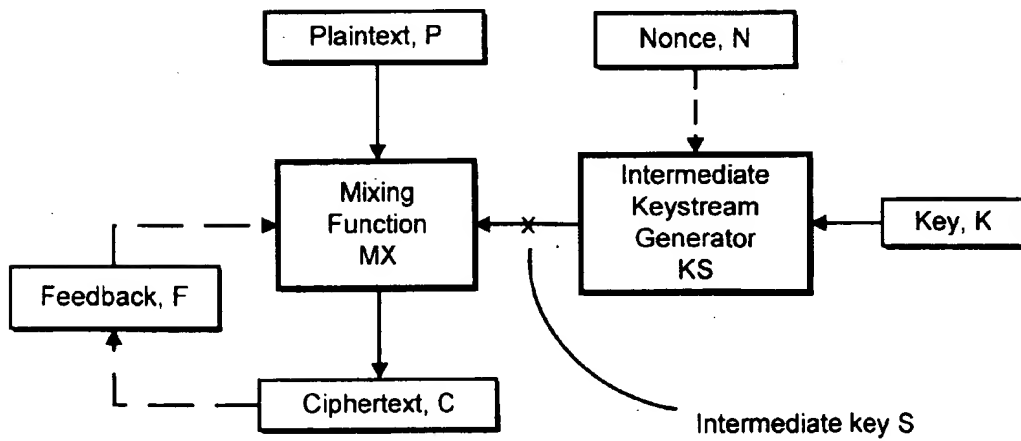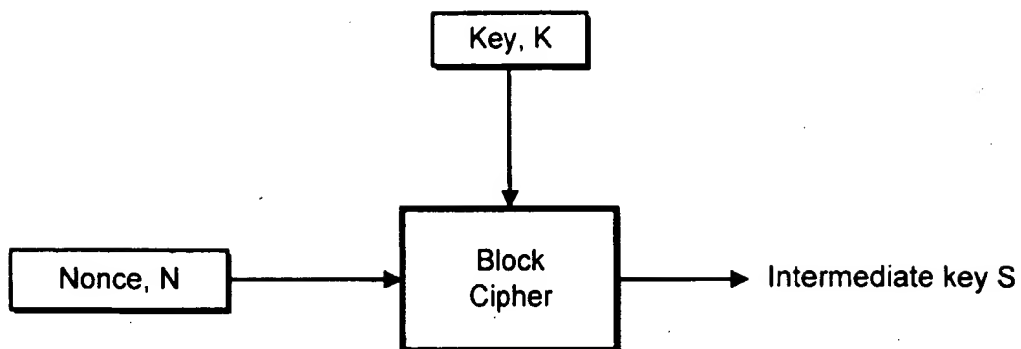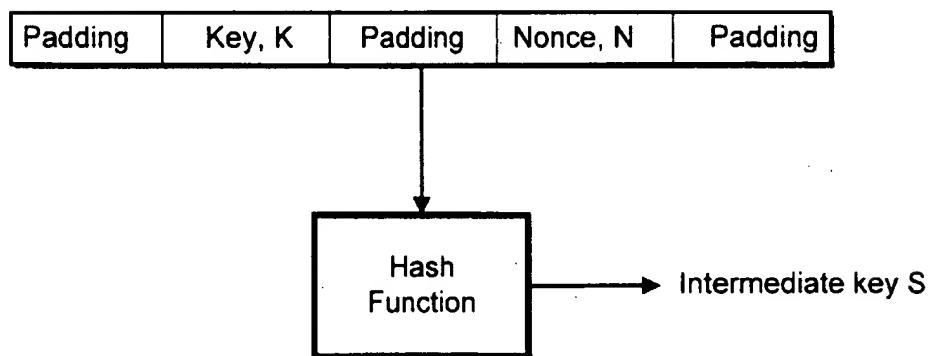
Figure 4

Figure 5

Figure 6

s1

S = KS(K) = S(1)S(2)S(3) ... S(n)

s2

C(i) = MX(P(i), S(i))

FOR i = 1 to n
REPEAT step s2

END ENCRYPTION

Figure 7

Figure 8

s3

$$S = KS(K) = S(1)S(2)S(3) ... S(n)$$
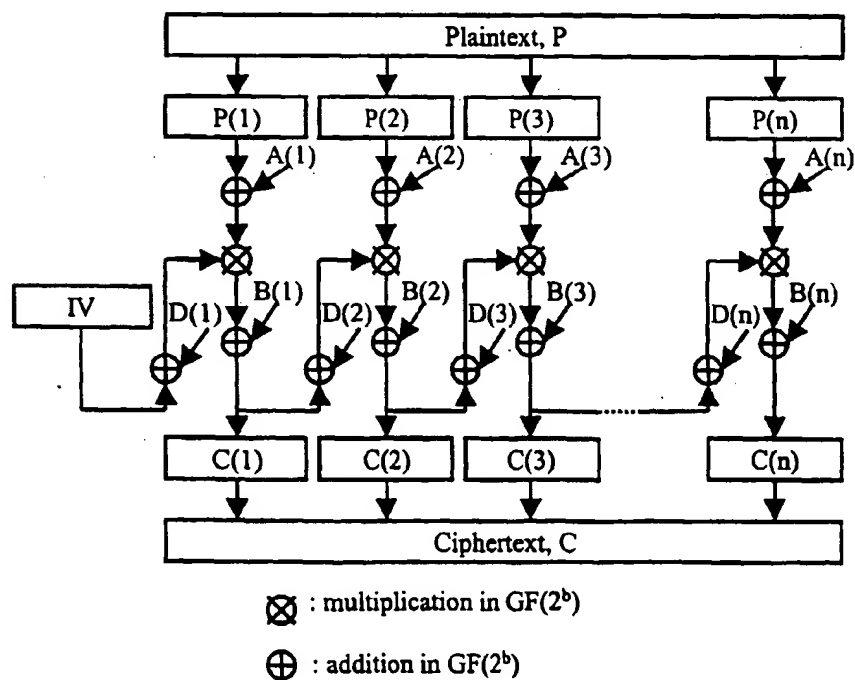
s4

$$P(i) = MX^{-1}(C(i), S(i))$$
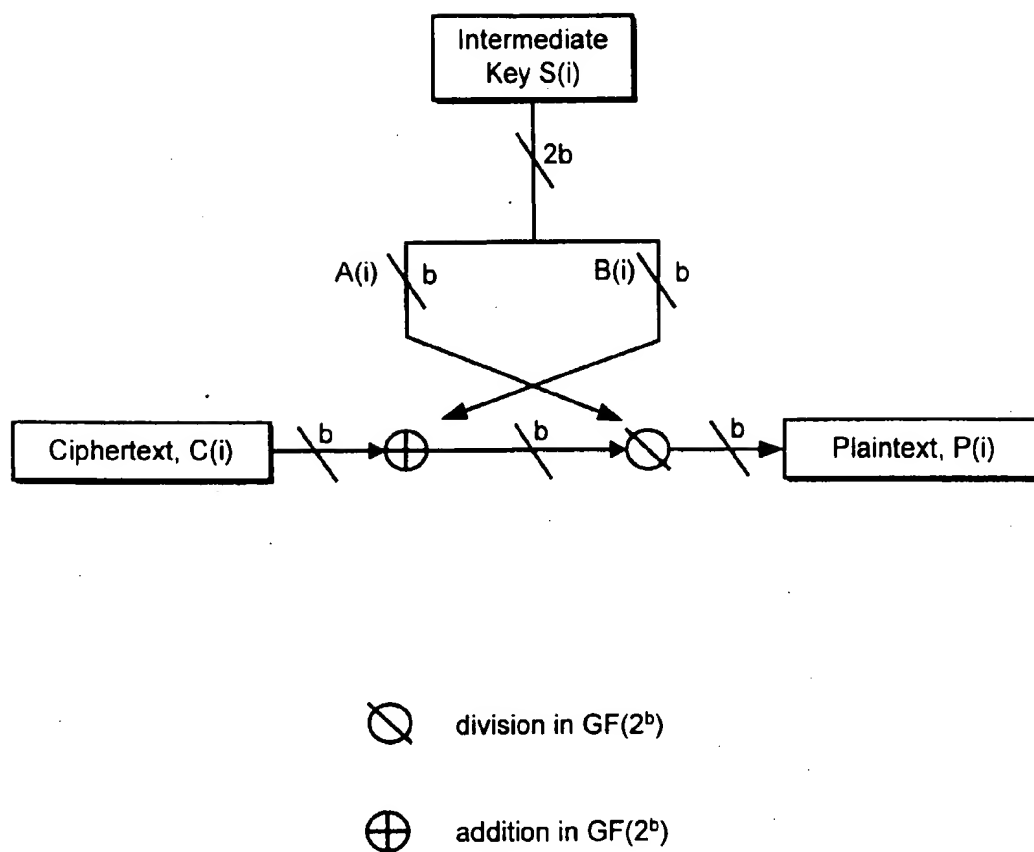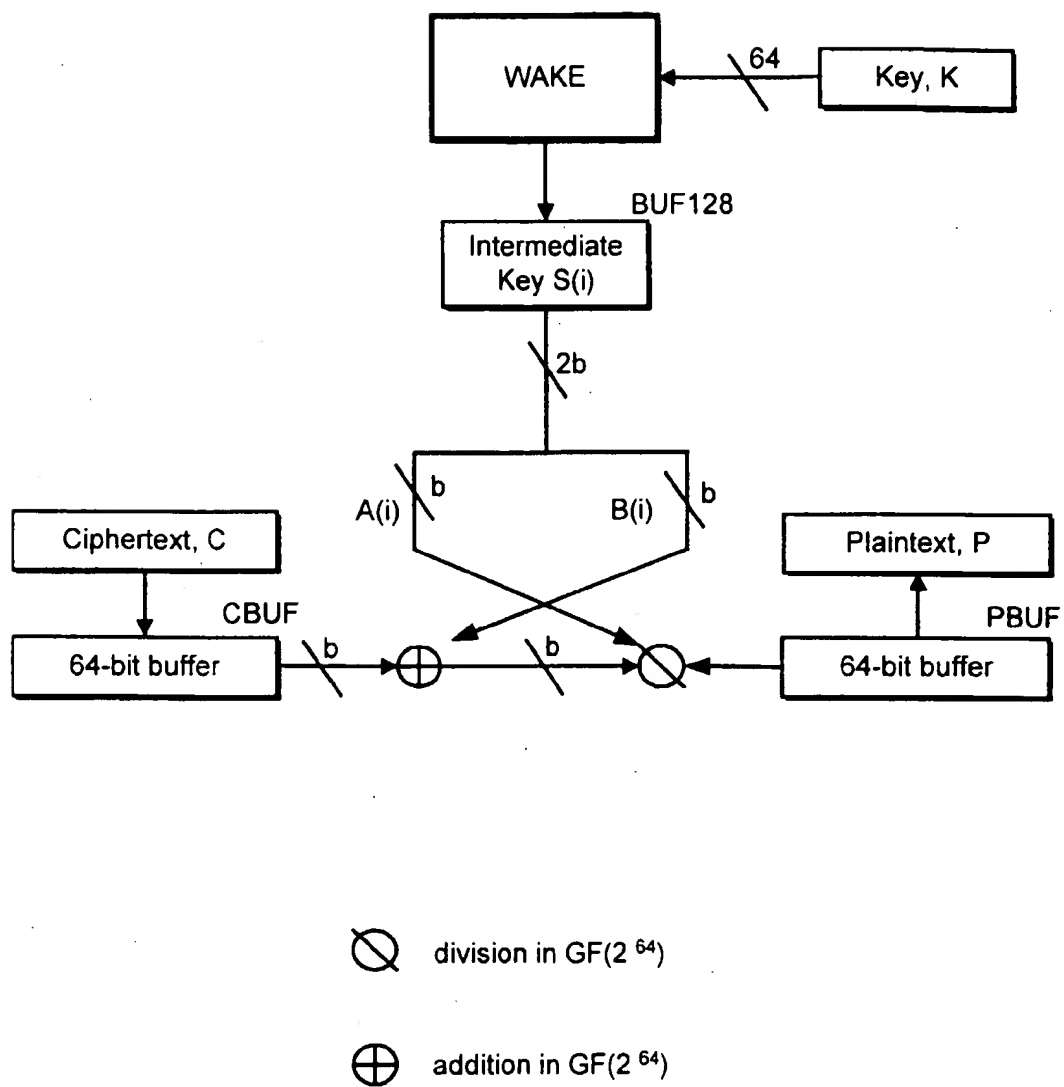
FOR i = 1 to n
REPEAT step s4

END DECRYPTION
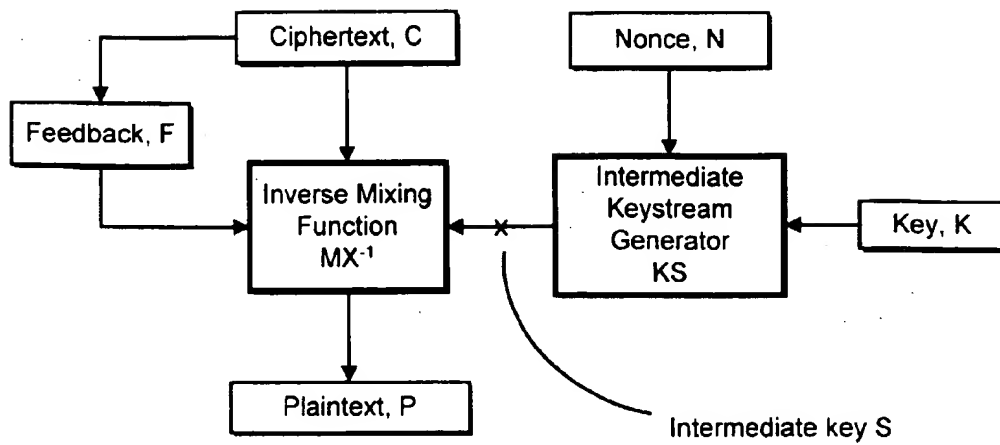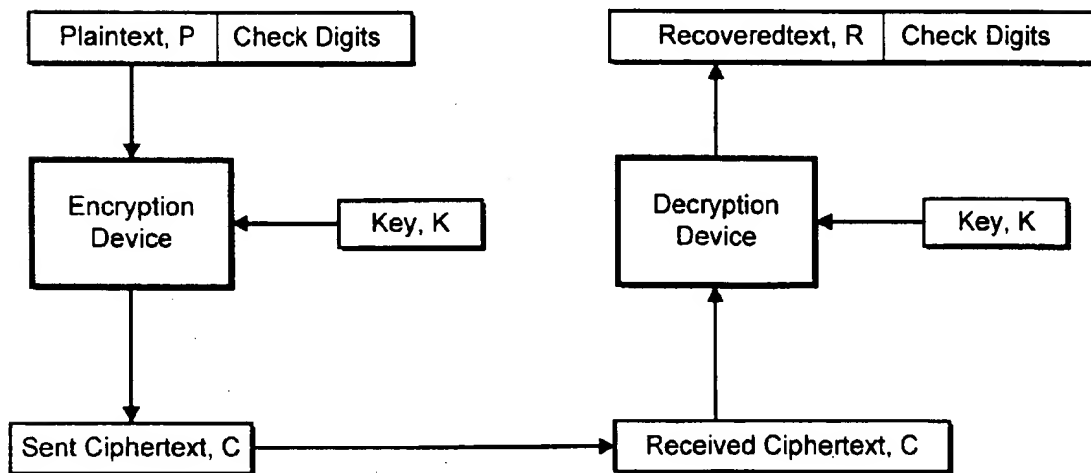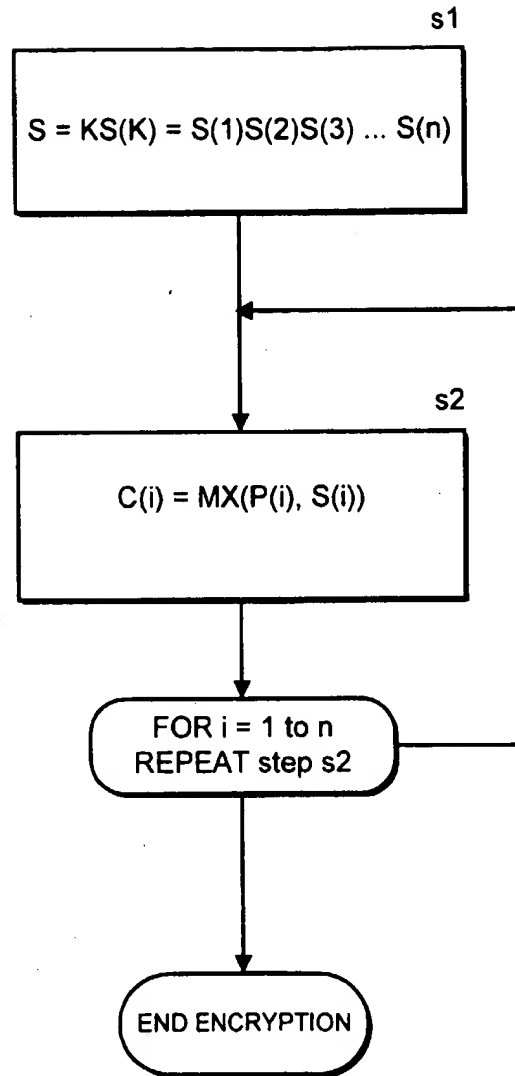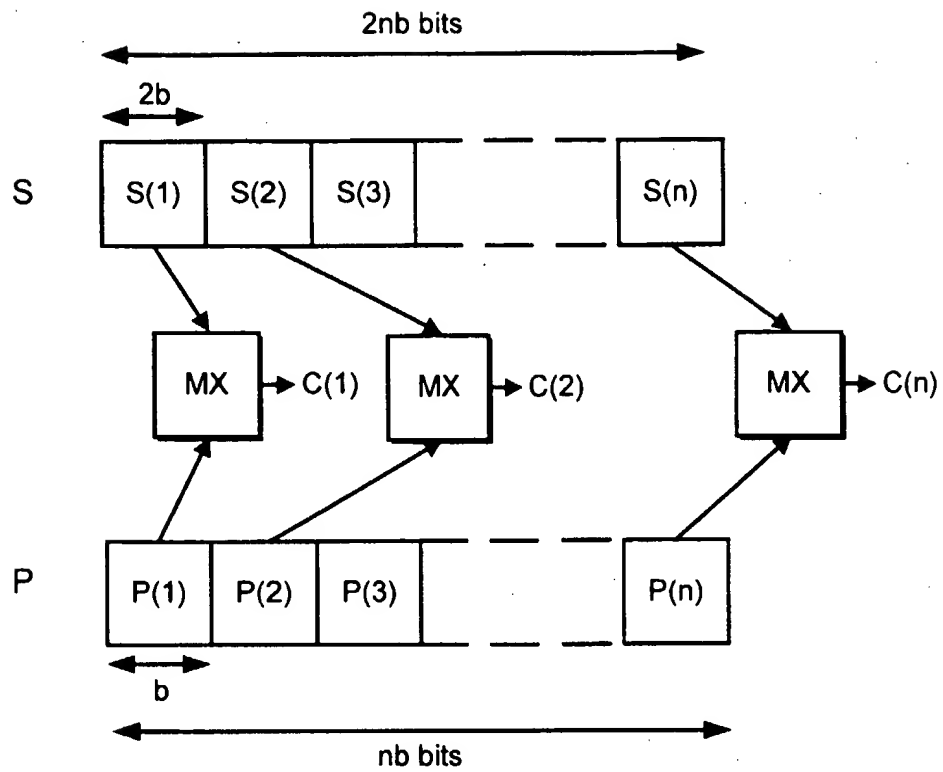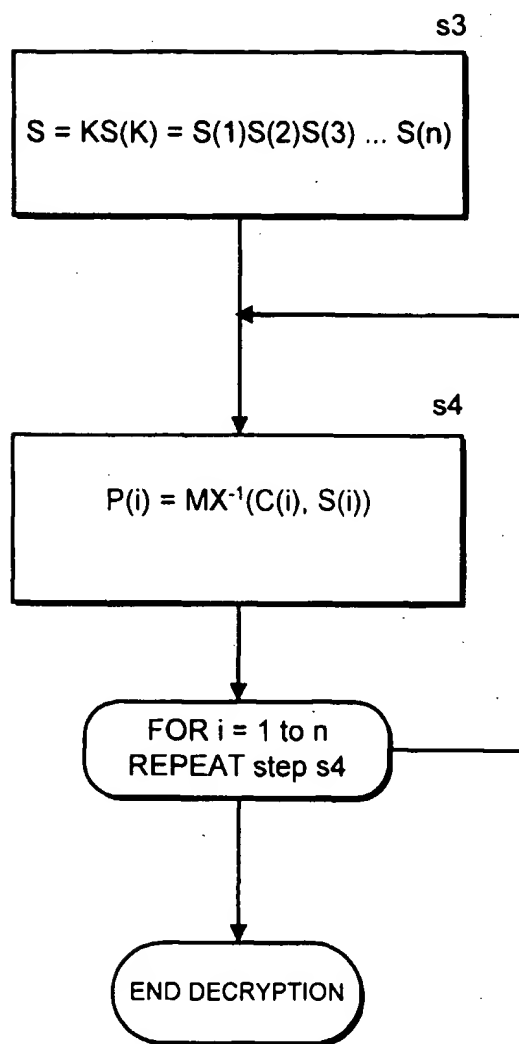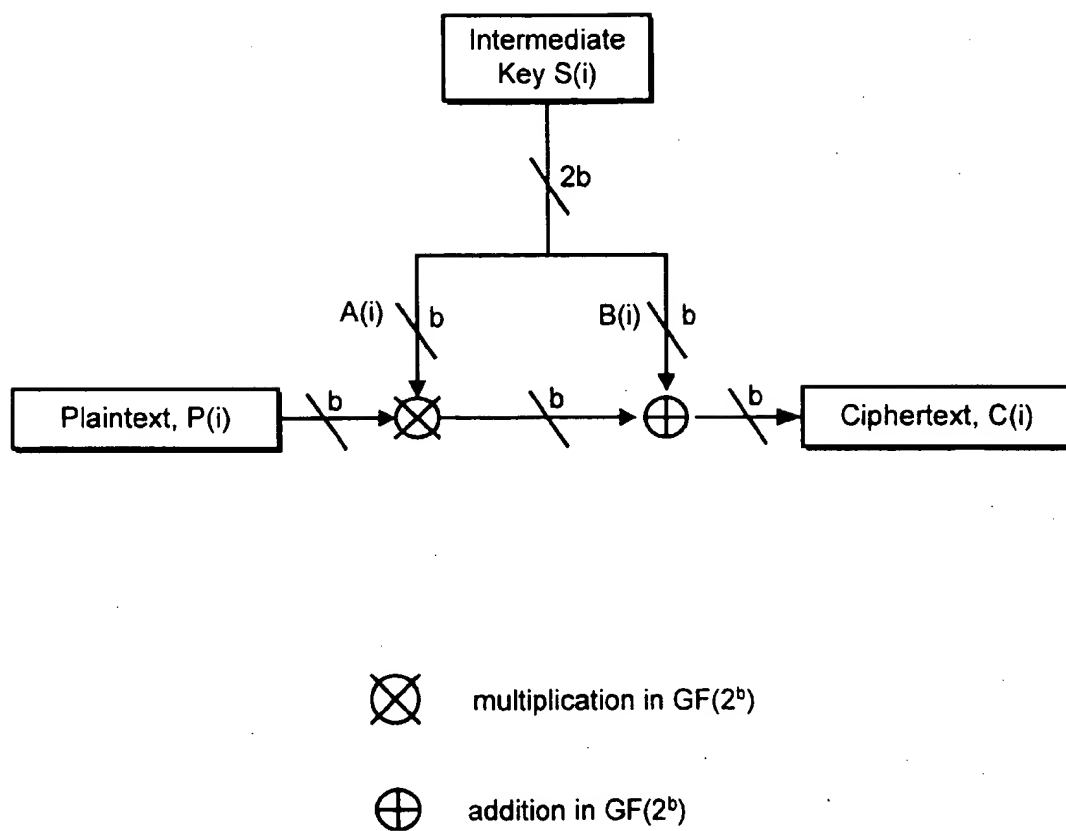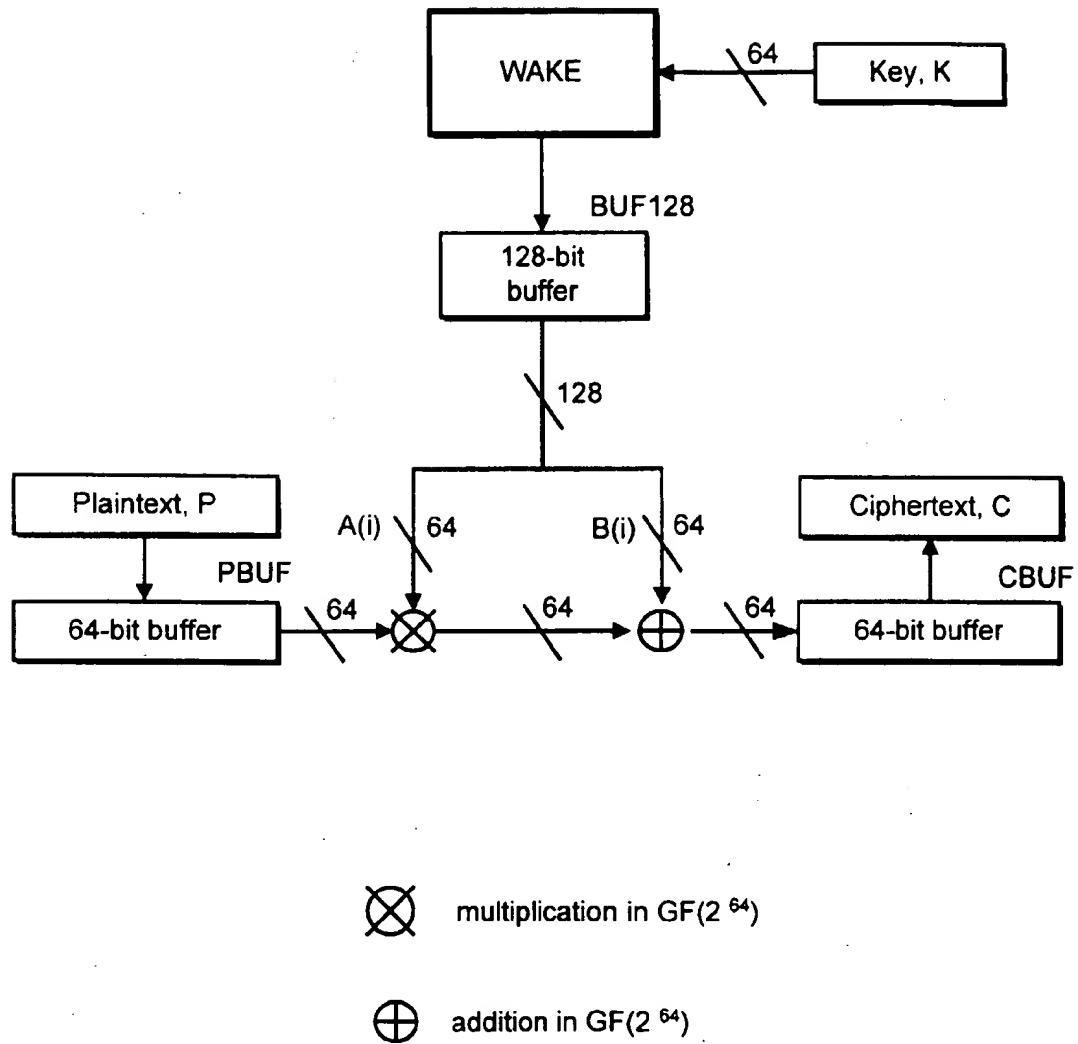
Figure 9

Figure 10

Figure 11

Figure 12

Figure 13

Plaintext, P

P(1) P(2) P(3) P(n)

B(1) B(2) B(3) B(n)

IV A(1) A(2) A(3) A(n)

C(1) C(2) C(3) C(n)

Ciphertext, C

$\bigotimes$: multiplication in GF(2^b)

$\bigoplus$: addition in GF(2^b)

Figure 14

Plaintext, P

P(1) P(2) P(3) P(n)

B(1) B(2) B(3) B(n)

Combine

encrypt

IV A(1) A(2) A(3) A(n)

C(1) C(2) C(3) C(n)

Ciphertext, C

$\bigotimes$: multiplication in GF(2^b)

$\bigoplus$: addition in GF(2^b)

integrity
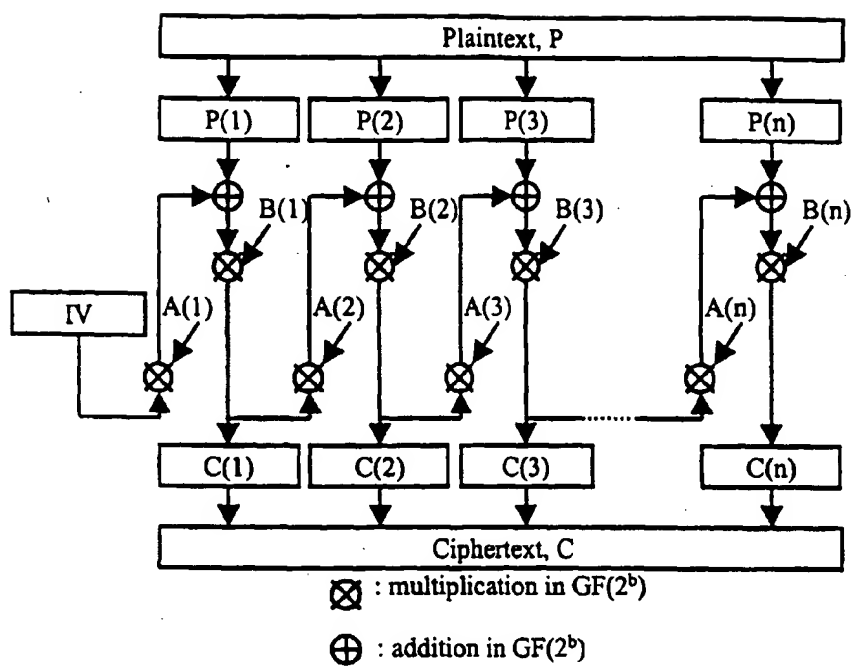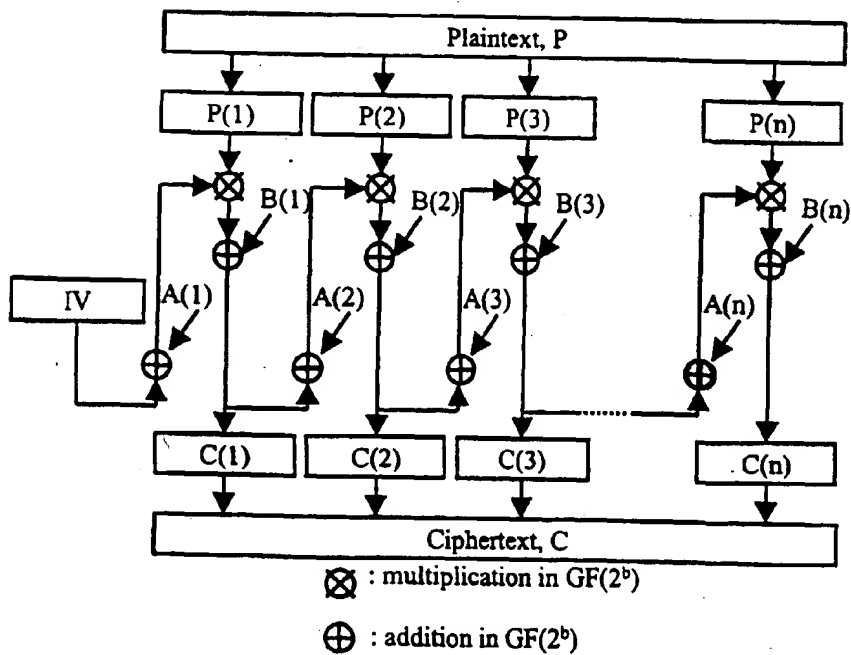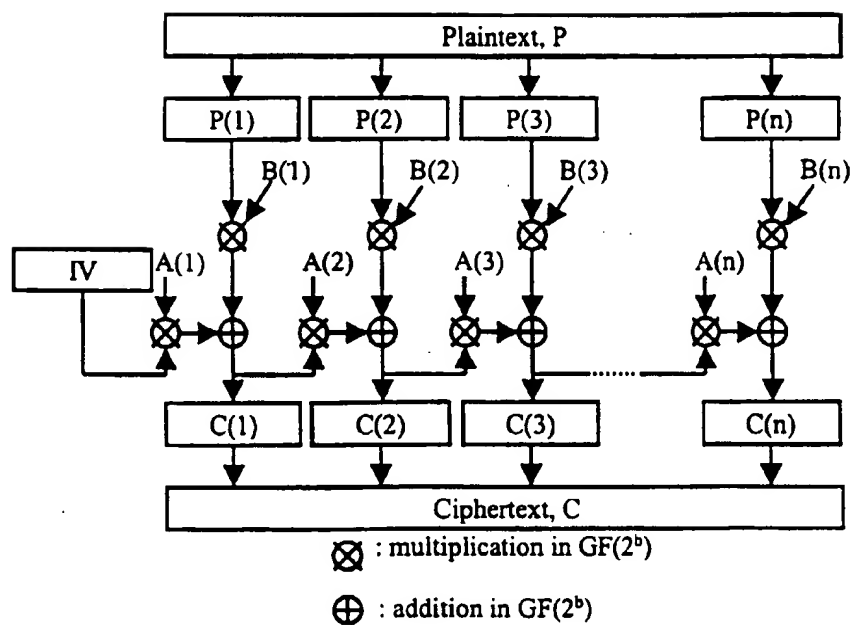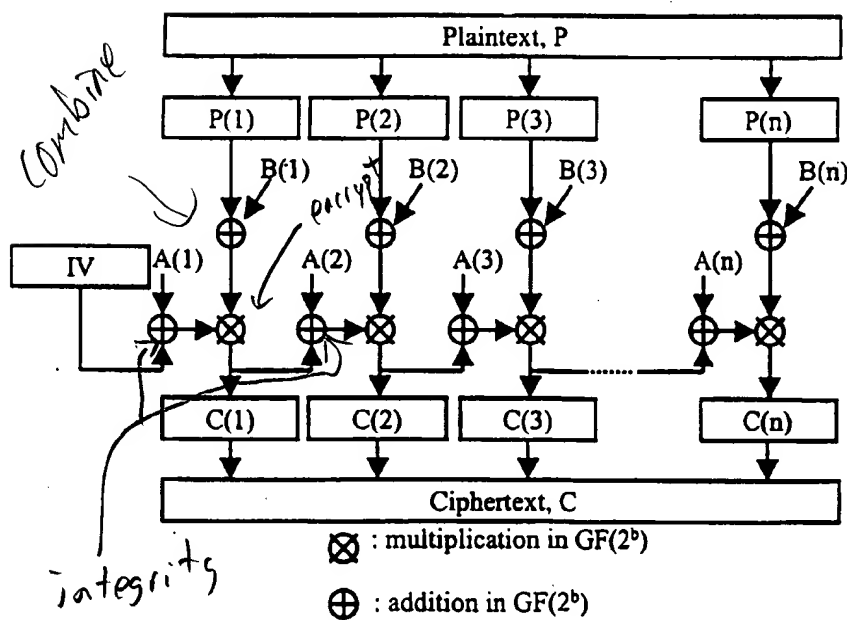
Figure 15

Figure 16

Figure 17

: multiplication in $GF(2^b)$

: addition in $GF(2^b)$

Figure 18

: multiplication in $GF(2^b)$

: addition in $GF(2^b)$

Figure 19

Figure 20

division in GF(2^{64})

addition in GF(2^{64})
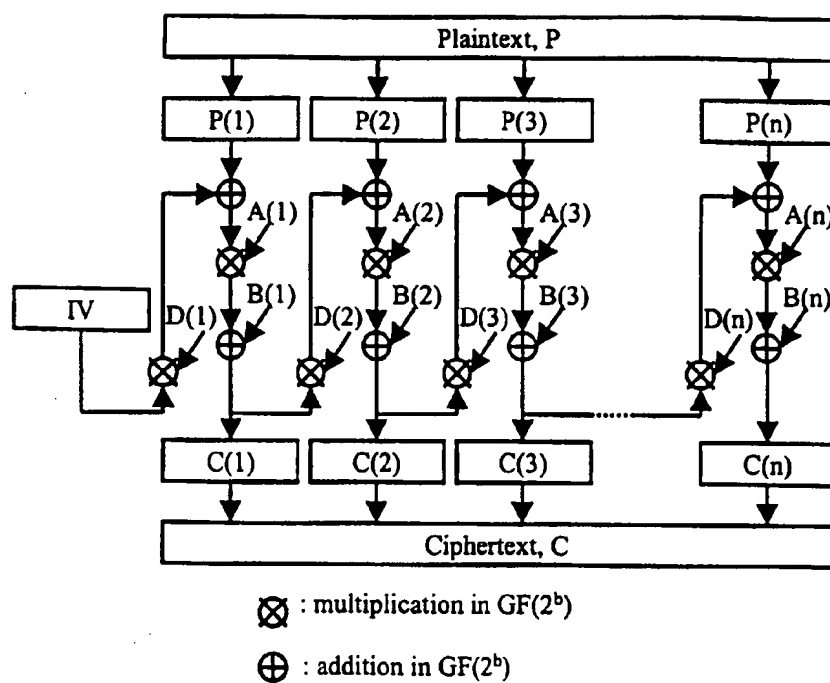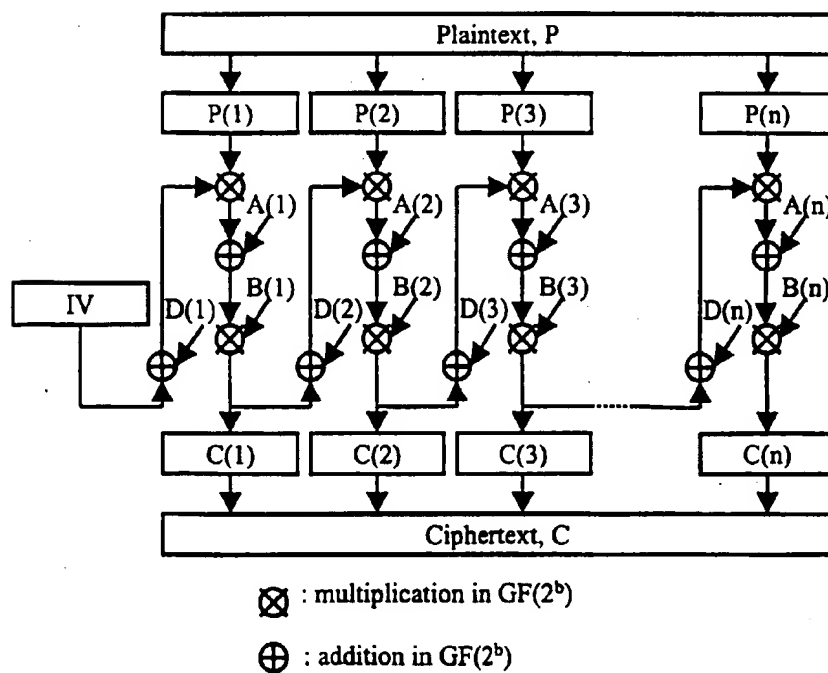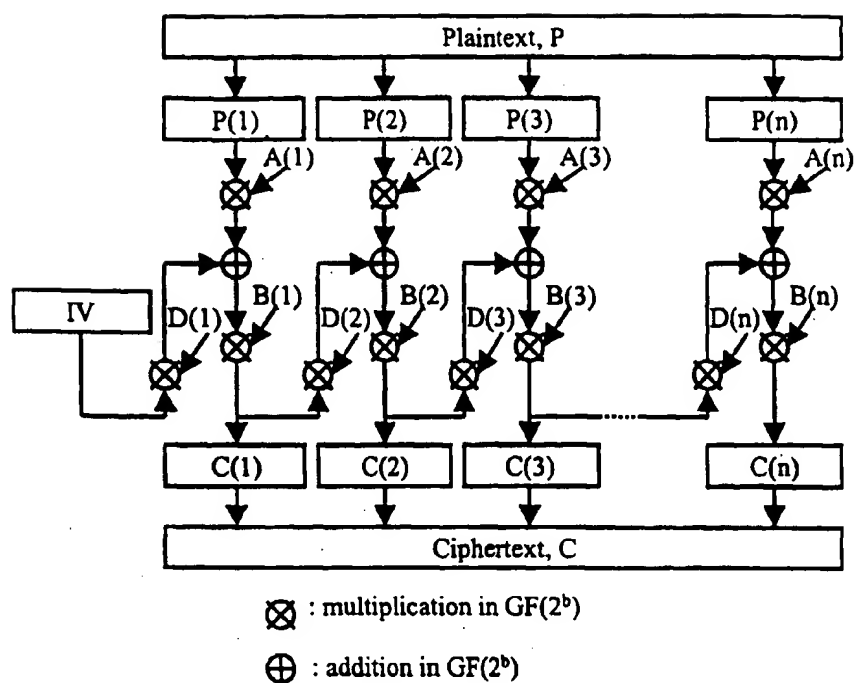
Figure 21

Figure 22

Figure 23

Nonce, N2

Nonce, N1

Intermediate
Keystream
Generator
KS2

×

Intermediate
Keystream
Generator
KS1

Key, K

Intermediate
key S2

Intermediate key S1

Figure 24

Nonce, N1

File data

(file name)(time stamp)(owner)

Mixing
Function
MX

KS2

KS1

Key, K

Encrypted file data

Intermediate key S2

File dependent key S1

Figure 25

CT 1

Counter

KS ← Key, K

P → MX → C → MX⁻¹ → P

CT 2

Counter

KS ← Key, K

Figure 26

Message

Key Data
& Counter
Sync

Key Exchange Protocol

Key Data
& Counter
Sync

Message

Code → CPU
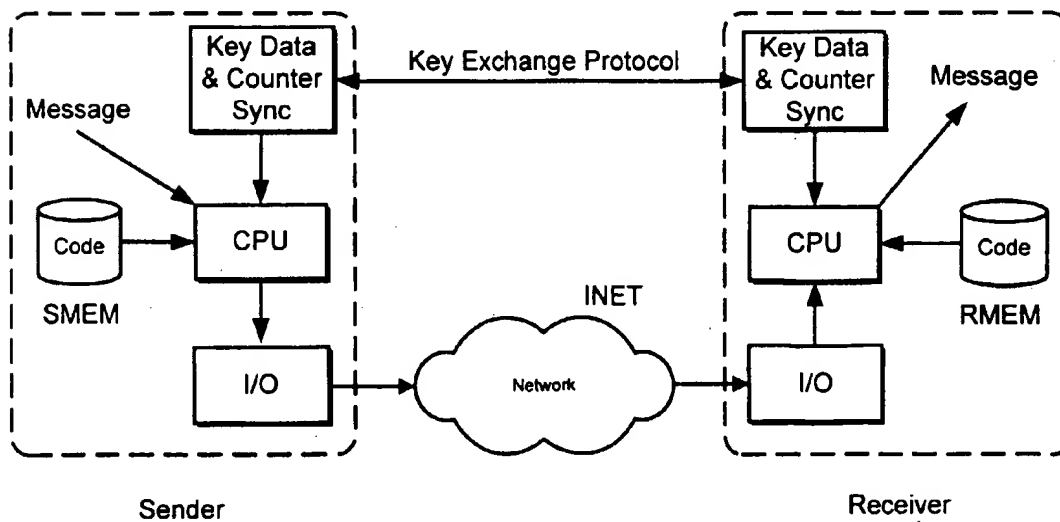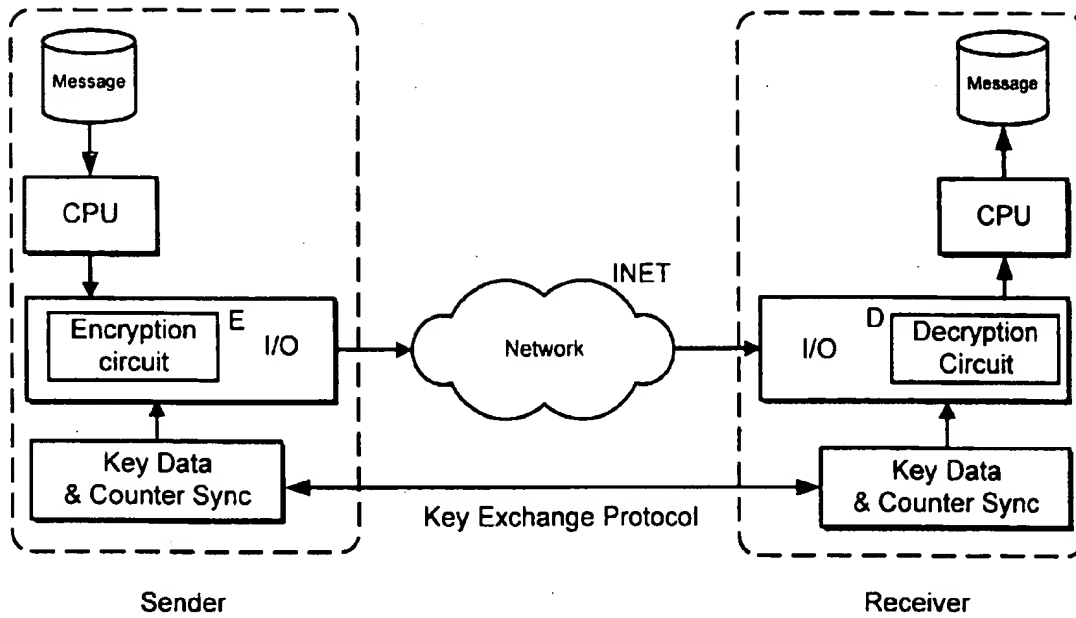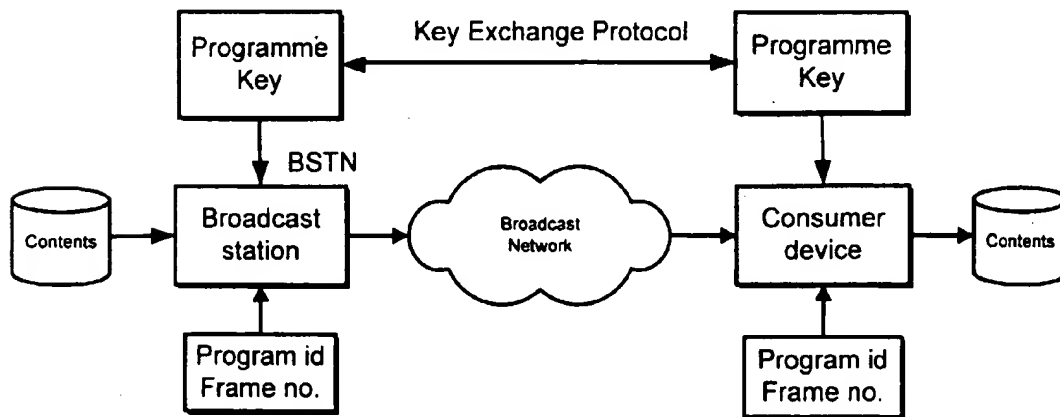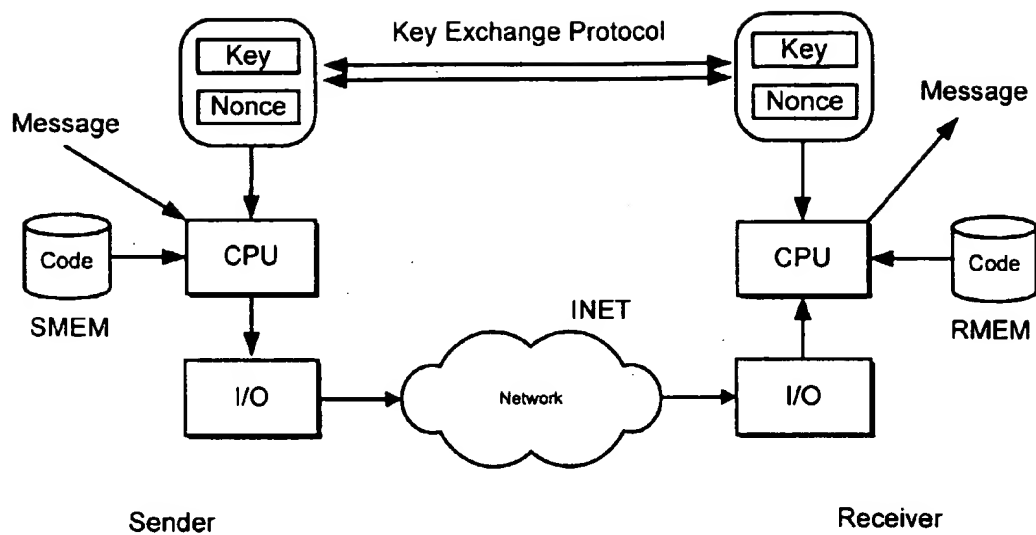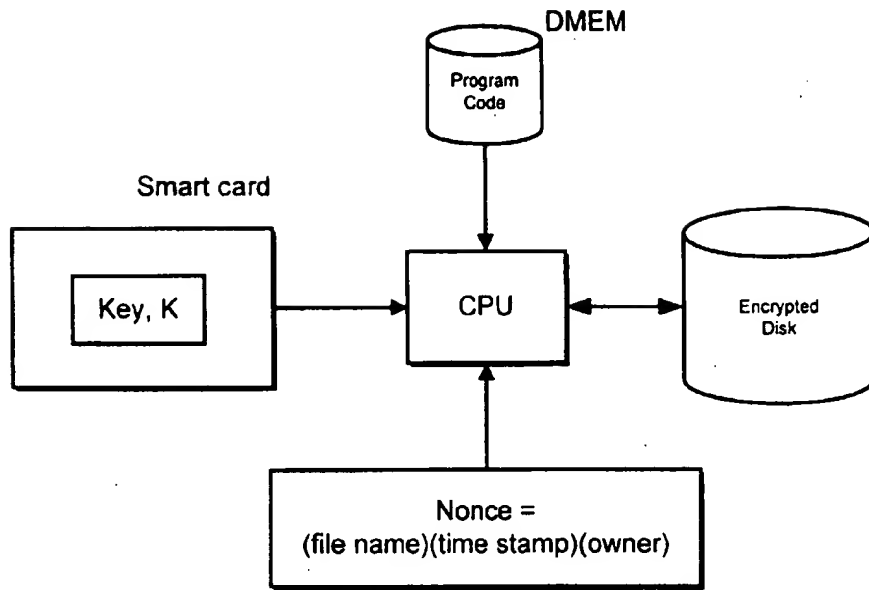
SMEM

I/O

INET

Network

CPU ← Code

RMEM

I/O

Sender

Receiver

Figure 27

Figure 28

Figure 29

Figure 30

DMEM

Program
Code

Smart card

Key, K

CPU

Encrypted
Disk

Nonce =
(file name)(time stamp)(owner)

Figure 31

DMEM

Program
Code

Smart card

Key, K

CPU

Database

Nonce =
(ID)(category)(rev.no./time stamp)

Figure 32

Figure 33

Nonce, N1

Programme

(programme name)(date)(version no.)

Mixing
Function
MX

KS2

KS1

Key, K

Intermediate key S2

Programme dependent
key S1

Figure 34

Programme

Inverse Mixing
Function
MX^{-1}

KS2

Programme dependent
key S1
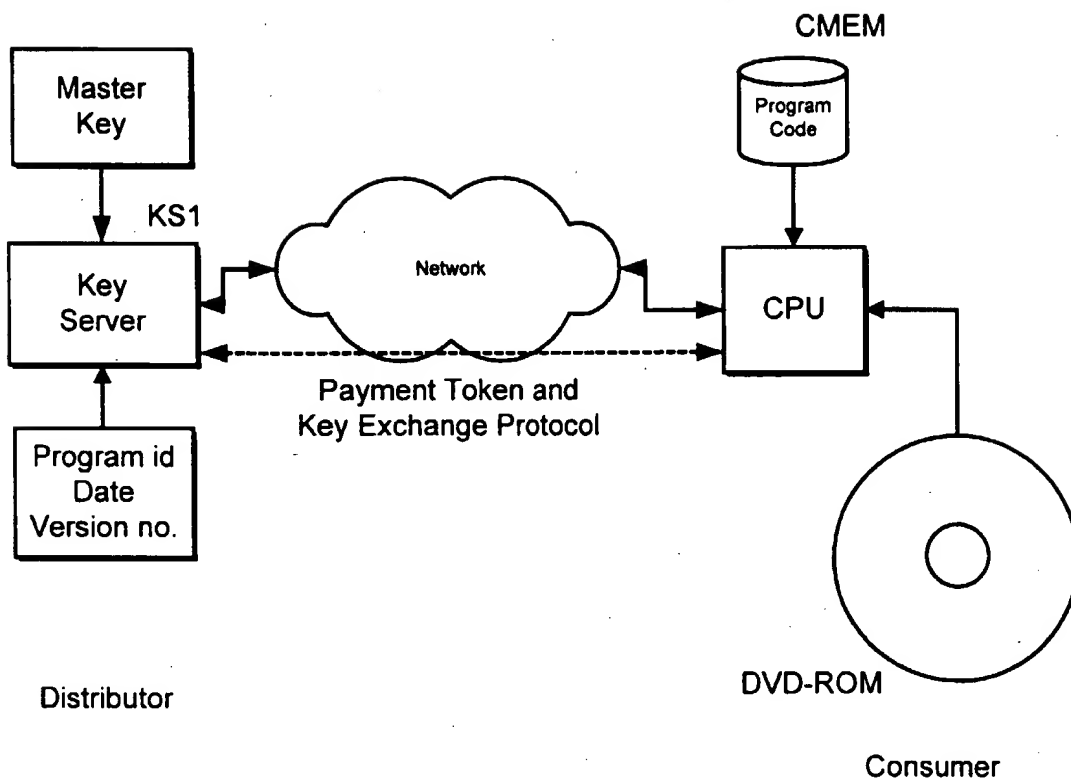
Intermediate key S2
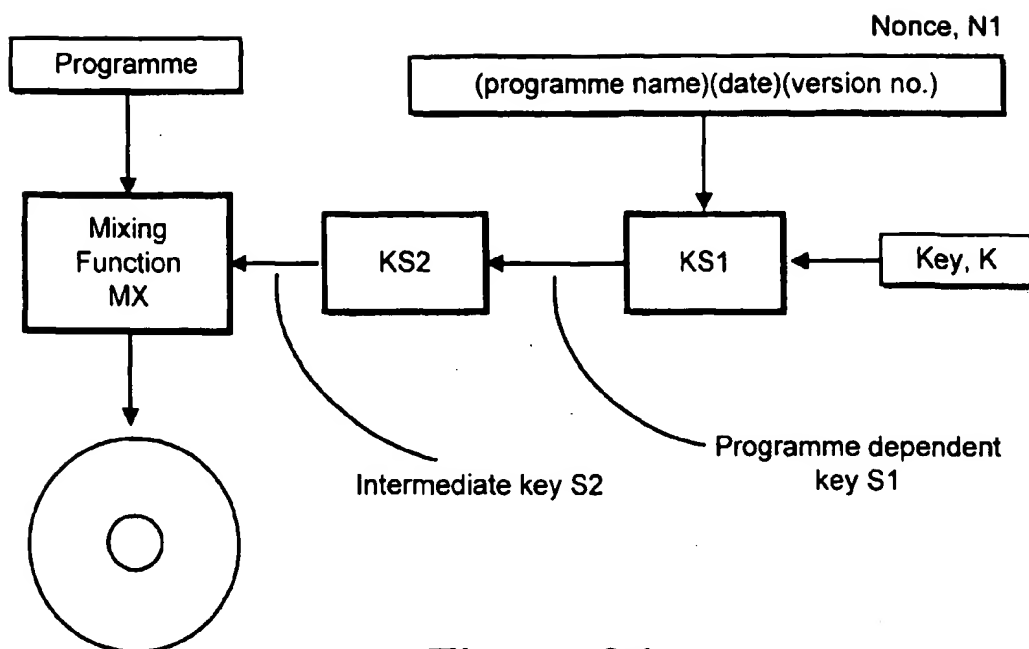
Figure 35
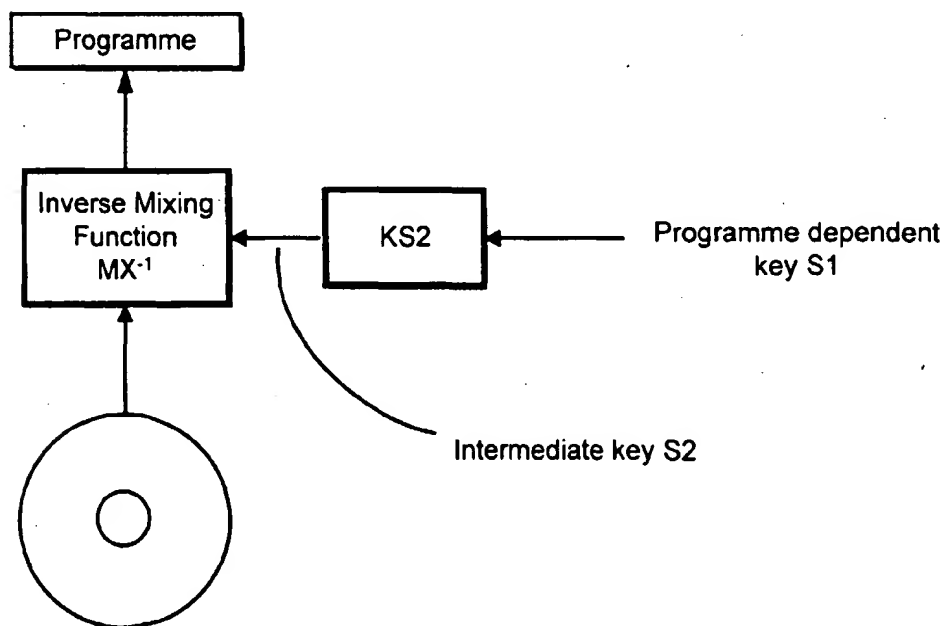
33

02/11/2004, EAST Version: 1.4.1
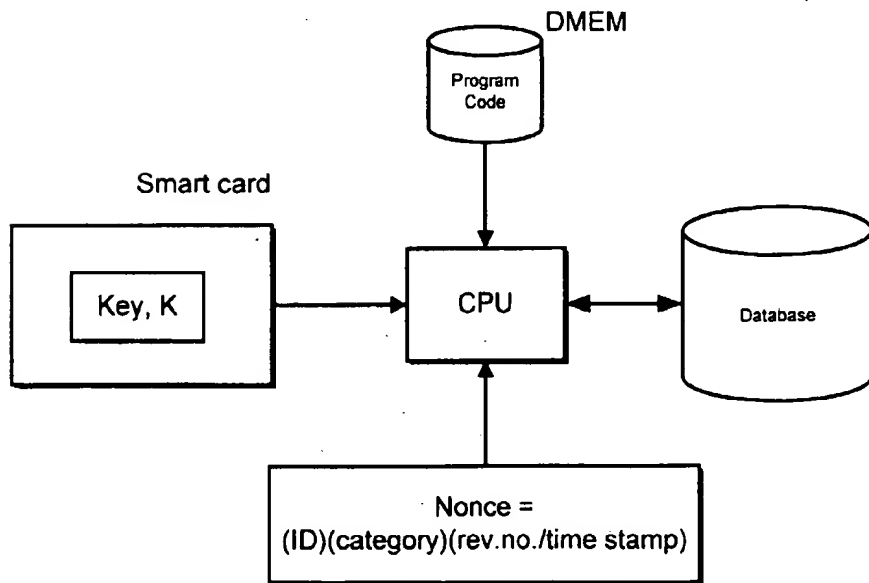
European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 99 30 4886

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
Y	"EFFICIENT, PARALLELIZABLE METHODS FOR DATA ENCRYPTION" IBM TECHNICAL DISCLOSURE BULLETIN,US,IBM CORP. NEW YORK, vol. 38, no. 3, page 75-77 XP000507984 ISSN: 0018-8689 * the whole document *	1-8, 11-15, 19-29	H04L9/06 H04L9/20
Y	BRUEGGERMANN T: "EIN SCHWEIZER SCHLUESSEL MIT WELTWEITER ZUKUNFT?" PROTECTOR,CH,GRAF UND NEUHAUS, ZURICH, vol. 21, no. 6, page 51-52,55,57,59 XP000414355 ISSN: 0256-4319 * page 55, column 3, line 39 - page 56, column 1, line 12 *	1-8, 11-15, 19-29	
Y	"CRYPTOGRAPHIC DEVICE" IBM TECHNICAL DISCLOSURE BULLETIN., vol. 16, no. 10, March 1974 (1974-03), pages 3406-3409, XP002122945 IBM CORP. NEW YORK., US ISSN: 0018-8689 * the whole document *	7,8,11, 12	TECHNICAL FIELDS SEARCHED (Int.Cl.7) H04L
A	EP 0 751 646 A (CANON KK) 2 January 1997 (1997-01-02) * column 10, line 40 - column 12, line 44 * * column 49, line 15 - column 50, line 25 * * figures 1-3 *	1,2,4, 19,20, 23-25, 27-29	

-/--

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	17 November 1999	Masche, C

EPO FORM 1503 03.82 (P04C01)

34

		Application Number
European Patent Office	**EUROPEAN SEARCH REPORT**	EP 99 30 4886

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
A	EP 0 443 752 A (GEN INSTRUMENT CORP) 28 August 1991 (1991-08-28) * abstract * * page 3, line 58 - page 5, line 10 * * page 5, line 41 - line 43 * -----	1,19,20, 24,27-29	
			TECHNICAL FIELDS SEARCHED (Int.Cl.7)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	17 November 1999	Masche, C

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

02/11/2004, EAST Version: 1.4.1

ANNEX TO THE EUROPEAN SEARCH REPORT
ON EUROPEAN PATENT APPLICATION NO.   EP 99 30 4886

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

17-11-1999

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
EP 0751646	A	02-01-1997	JP	9018468 A	17-01-1997
			JP	9018469 A	17-01-1997
			JP	9016678 A	17-01-1997
			JP	9016679 A	17-01-1997
			JP	9046329 A	14-02-1997
			AU	5619896 A	09-01-1997
			CA	2179971 A	31-12-1996
			CN	1142718 A	12-02-1997
EP 0443752	A	28-08-1991	US	5054067 A	01-10-1991
			AT	130144 T	15-11-1995
			AU	634857 B	04-03-1993
			AU	7100291 A	22-08-1991
			CA	2036489 A	22-08-1991
			DE	69114324 D	14-12-1995
			DE	69114324 T	20-06-1996
			IE	69271 B	21-08-1996
			JP	4215689 A	06-08-1992
			NO	303257 B	15-06-1998

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

02/11/2004, EAST Version: 1.4.1